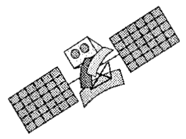
The Impact of Autonomous Systems Technology on JPL Mission Software

Dr. Richard J. Doyle

Center for Space Mission Information and Software Systems, and
Information Technologies and Software Systems Division
Jet Propulsion Laboratory
California Institute of Technology

24th Annual Software Engineering Workshop
Software Engineering Laboratory
NASA Goddard Space Flight Center



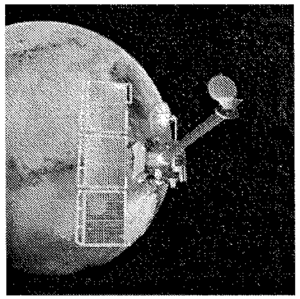
December 1, 1999

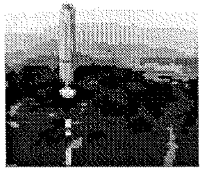
RJD.SEL99.1

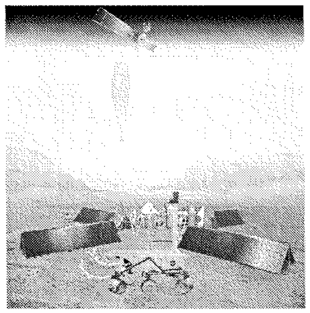
Autonomy for Future Missions

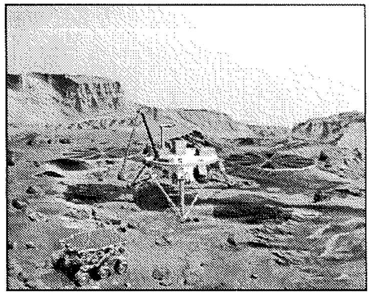
Mars Outposts

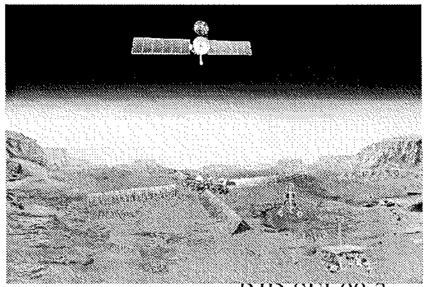
- **Remote Science Laboratories**
  - Tele-operated or autonomous laboratories in the planetary environment for handling and conducting in situ scientific investigations on collected samples
- **Three scales / resolutions**
  - remote sensing
  - distributed sensing
  - point sensing
- **Heterogeneous, cooperating networks**
  - distributed networks of sensors, rovers, orbiters, permanent science stations, probes: all of which respond to sensing events, discoveries, changing PI directions, etc., to provide rich presence in Mars environment for science community and public
- **Infrastructure**
  - Planetary permanent infrastructure to support series of science and/or commercial missions leading to human presence



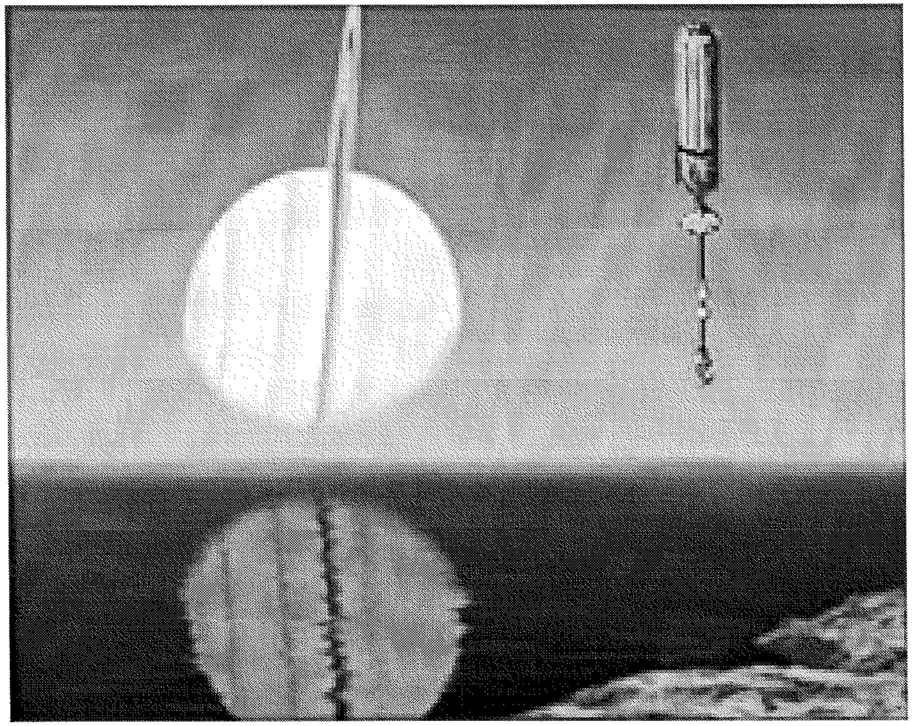






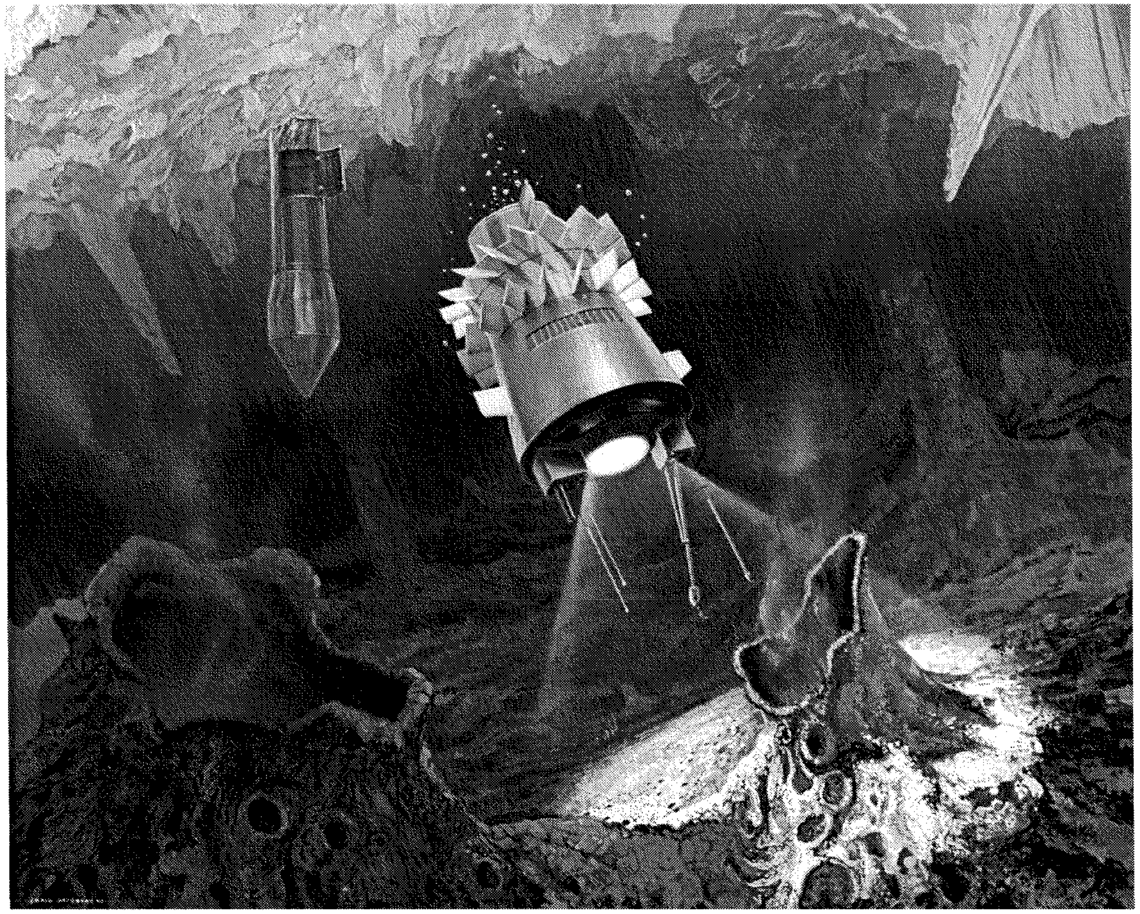


Titan Aerobot



- The aerobot conducts in-situ science operations when landed, and wide-area imaging when aloft.
- Archived and learned models of wind patterns assist path planning, enabling near-returns to areas of high scientific interest.

Europa Cryobot / Hydrobot

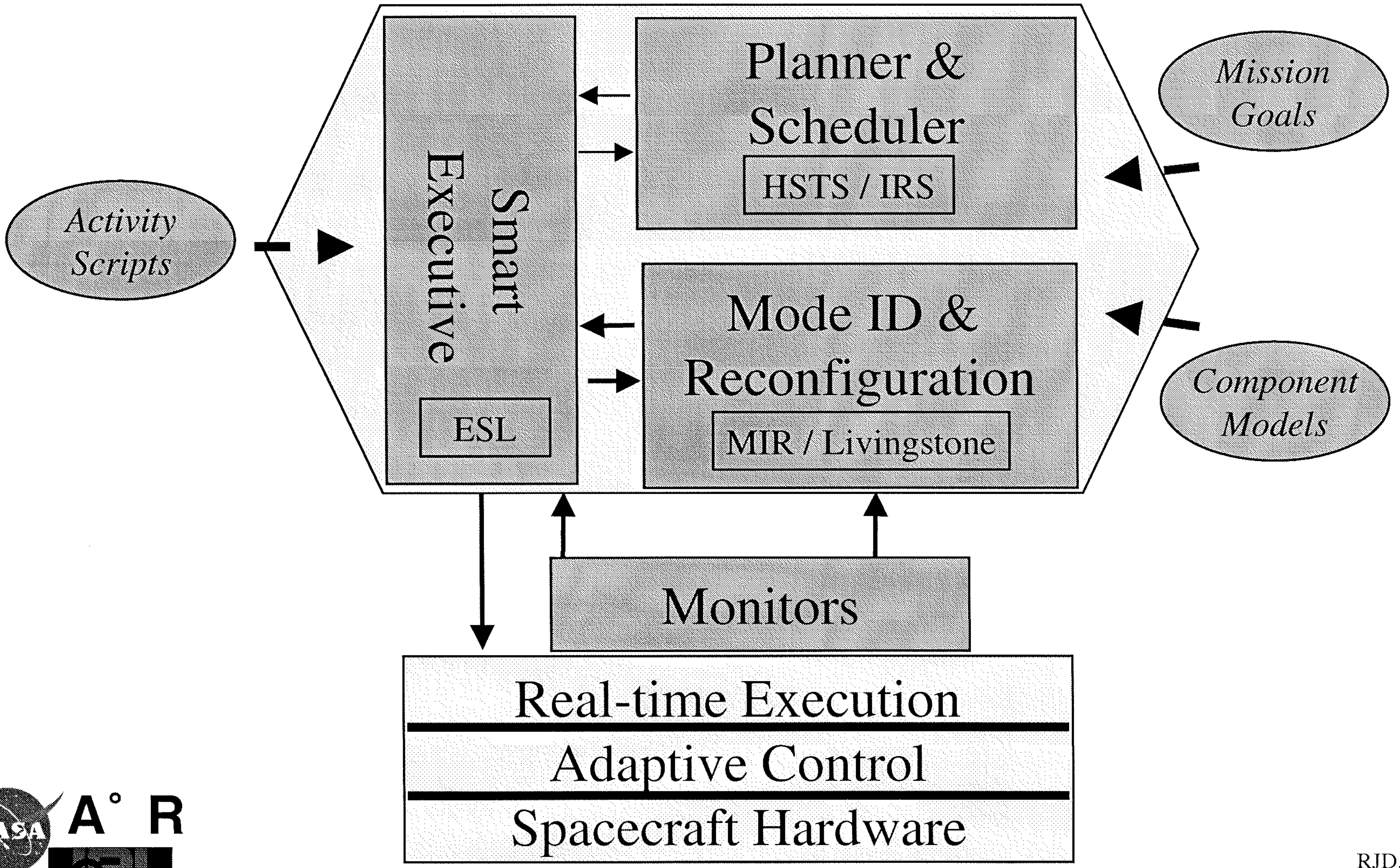


- Perhaps more than any other, a mission of discovery in a truly alien environment: How to know what to look for? How to recognize it?

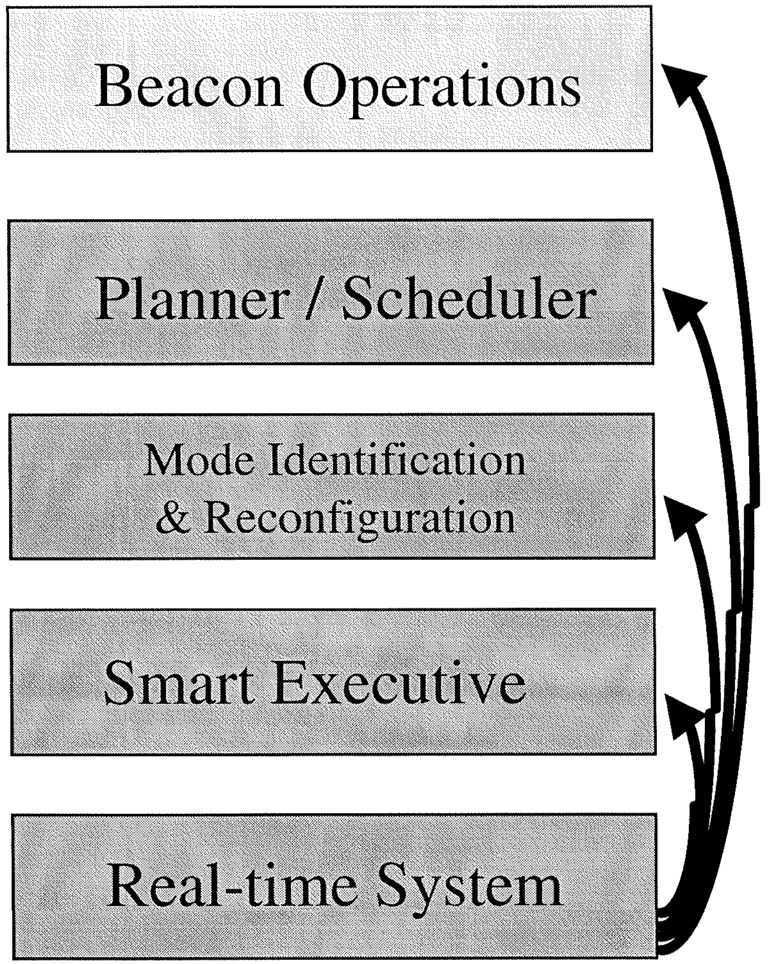
The Emergence of Autonomy

D. Bernard, P. Nayak et al

Remote Agent Architecture



Closing Loops Onboard



Ground assistance invoked with focused report on spacecraft context and history

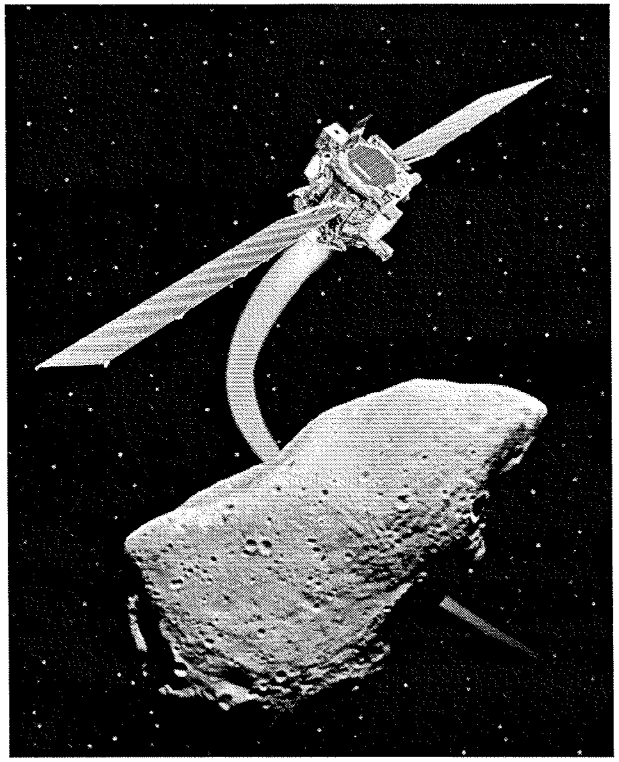
Replanning of mission activities around altered resources or functions

Diagnosis of faults and informed selection of recovery actions

Local retries or alternate, pre-defined activities to achieve same goal

Several layers of onboard recovery provides for unprecedented robustness in achieving mission goals in the face of uncertainty

New Millennium Flight Experiment

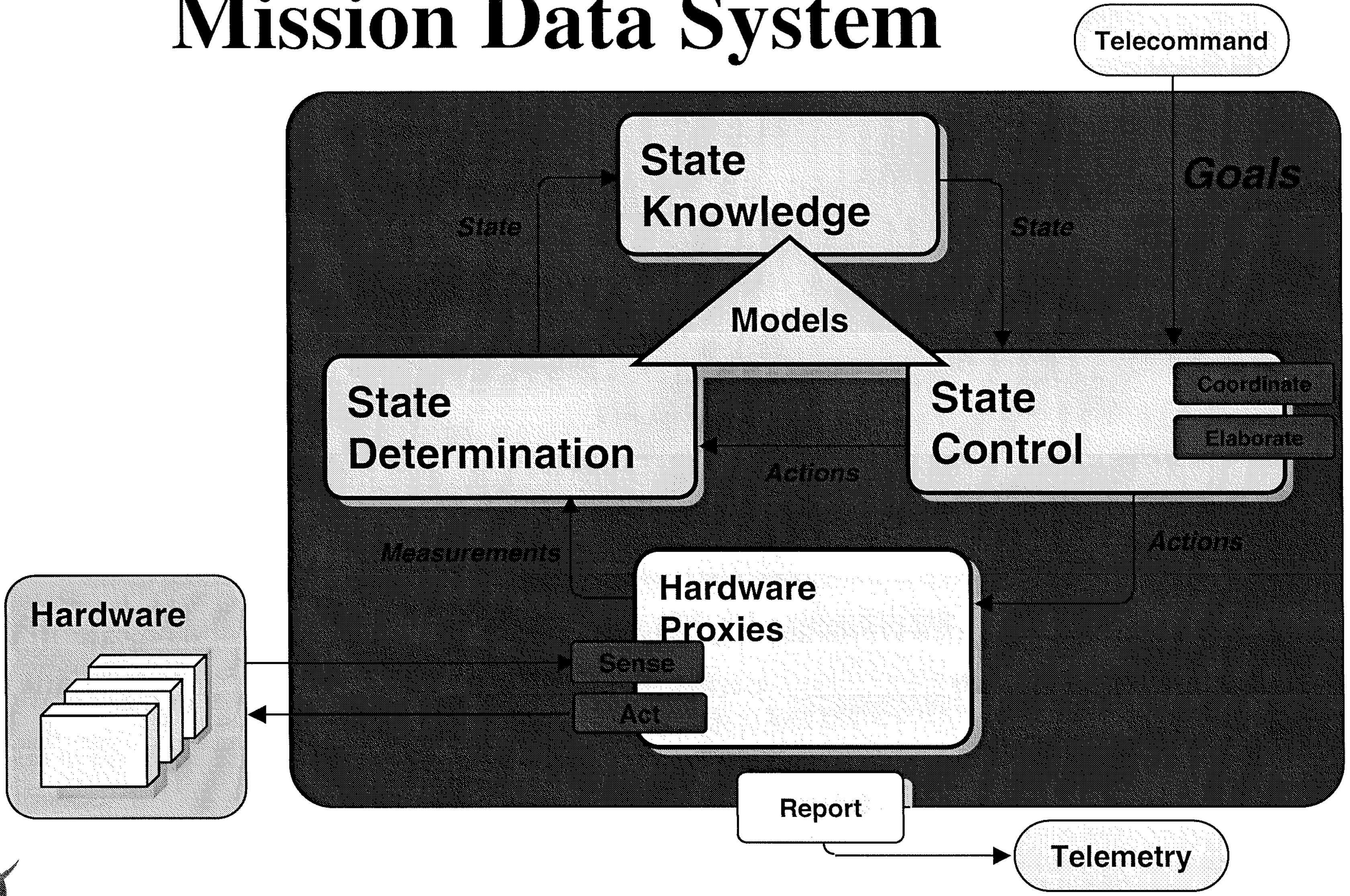


- DS-1 has encountered an asteroid and will encounter a comet.
- Remote Agent Experiment (RAX) achieved 100% of its technology demonstration goals in May '99.
- RAX joined eleven other DS-1 technology experiments such as onboard optical navigation and solar electric propulsion.

Software Engineering Challenges

R. Rasmussen et al

Influence of Remote Agent: Mission Data System



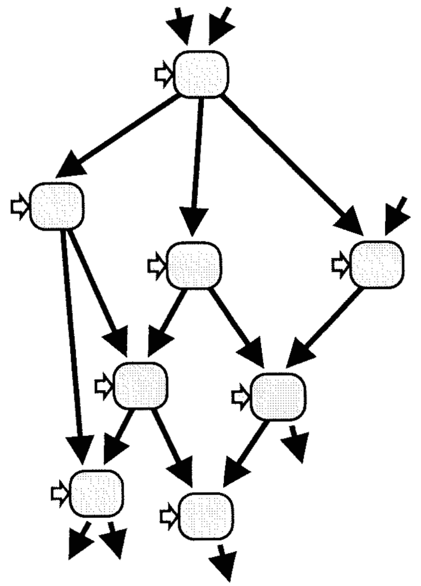
MDS Architectural Themes

- Unifying state-based paradigm behind all elements
- Extensive and explicit use of models
- Goal-directed operations specifies intent, simplifies workload
- Closed-loop control enables opportunistic science gathering
- Fault protection is natural part of robust control, not an add-on
- Explicit resource management (power, propellant, memory, etc)
- Navigation and attitude control build from common base
- Clean separation of state determination from control
- State uncertainty is acknowledged & used in decision-making
- Clean separation of data management from data transport
- Upward compatibility through careful design of interfaces
- Object-oriented components, frameworks, design patterns

Theme: Goal-Directed Operation

Definition

- A goal specifies *intent*, in the form of *desired state*.
- A *goal* is a constraint on the value of a state variable during a time interval.
- Goal-directed operation is simpler because a goal is easier to specify than the actions to accomplish it.
  - Goal-achieving modules (GAMs) attempt to accomplish submitted goals.
  - A GAM may issue primitive commands and/or sub-goals to other GAMs.
  - A GAM must either accomplish a goal or responsibly report that it cannot.



Scalable Autonomy

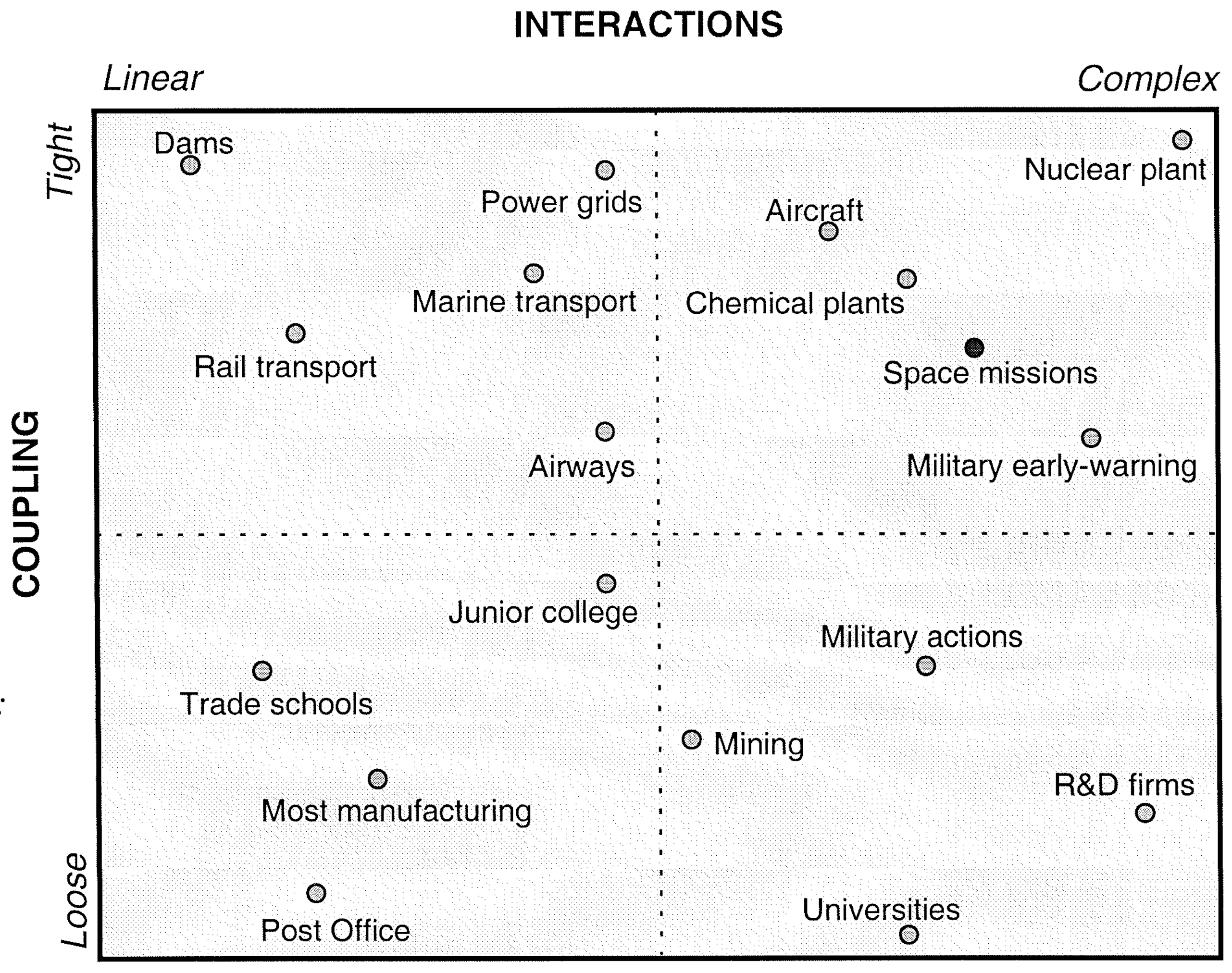
Capability	*Baseline* ⟷	*Greater Autonomy*
Planning & Scheduling	Plans generated and validated on ground; some automation	Plans generated onboard from uplinked goals within s/c and environment context
Execution	Highly predictable sequences fully compiled to a timeline	Flexible, deferred commanding; multi-threaded execution; local recovery and cleanup
Fault Protection	Fault identification puts spacecraft in safe hold; mission suspended	Model-based diagnosis and recovery for overall s/c state; mission continuation enabled

Model-based Lifecycle

- Model-based design and development
  - Models used across traditional subsystem domains: ACS, Navigation, Telecom, Thermal, Science Instruments, etc.
  - Models used across software domains: planning, execution, fault protection, data management, data transport, testing, etc.
  - Models used across the lifecycle: architecture, design, implementation, integration, test, delivery, operations, maintenance, reuse

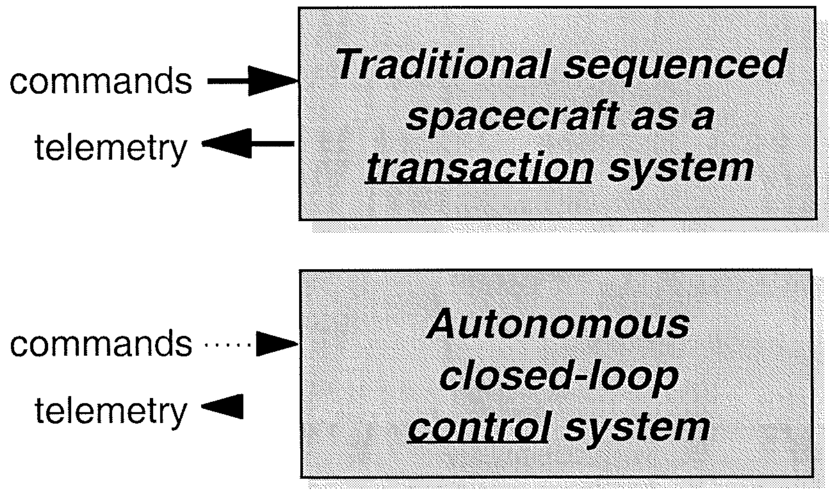
C. Perrow

Autonomy Software Validation



from
Normal Accidents: Living with High-Risk Technologies

Autonomy Software Validation


commands
telemetry
Traditional sequenced spacecraft as a <u>transaction</u> system
commands
telemetry
Autonomous closed-loop <u>control</u> system

Send command sequence.

Does the resulting telemetry match predictions?

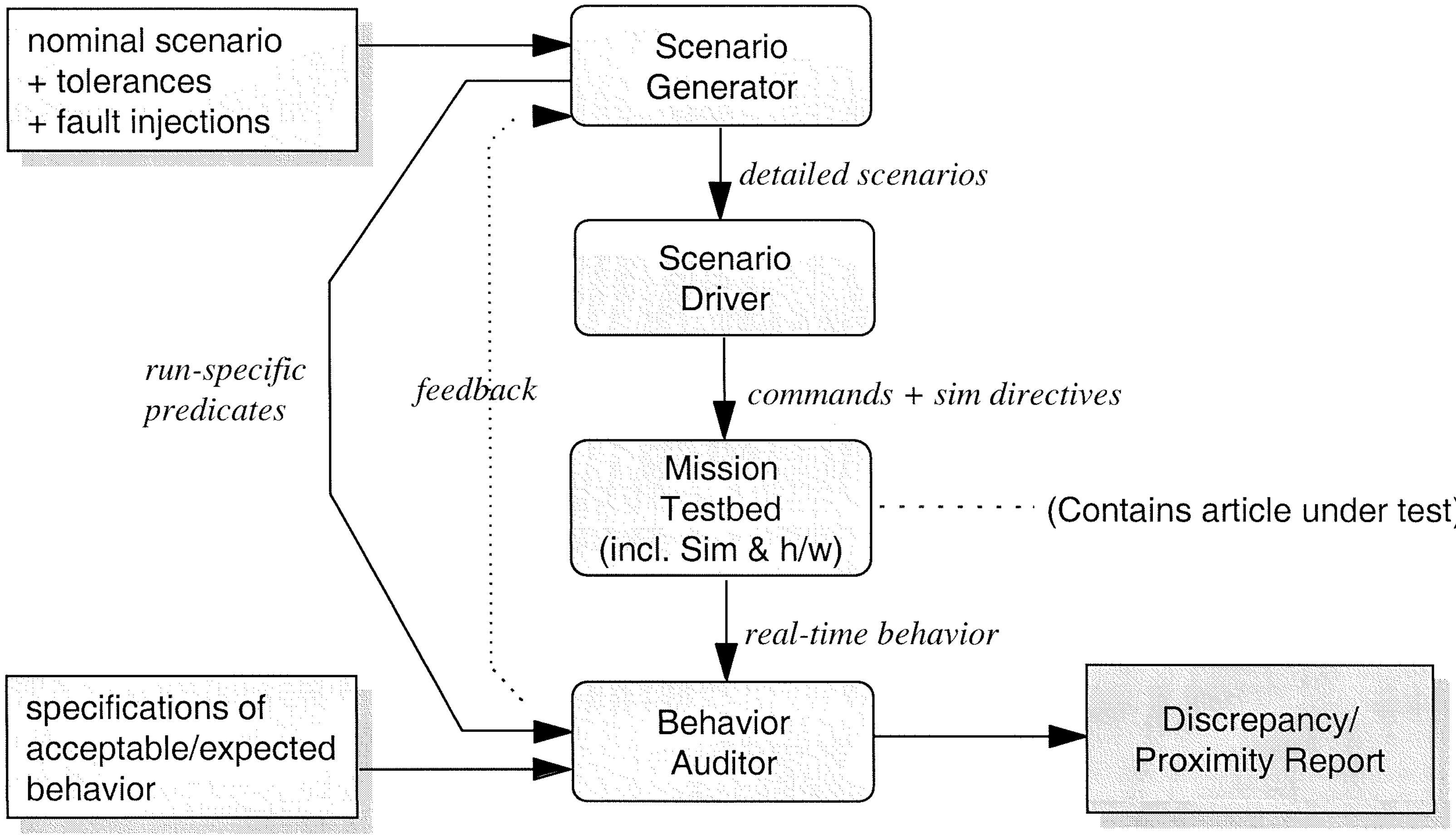
Observe behavior in the loop.

Is the system accomplishing its goals and respecting flight rules in spite of anomalies?

<u>Key idea</u>: (borrowed from model-based fault diagnosis)

- Do not attempt to enumerate all possible software failures
- Rather, define and identify departures from acceptable bounds on software behavior
- Apply at design, test and run time

Autonomy Software Validation


nominal scenario
+ tolerances
+ fault injections
Scenario Generator
detailed scenarios
Scenario Driver
commands + sim directives
Mission Testbed (incl. Sim & h/w)
(Contains article under test)
real-time behavior
run-specific predicates
feedback
specifications of acceptable/expected behavior
Behavior Auditor
Discrepancy/ Proximity Report

Analytic Verification Technology

(Automated Software Engineering Group, NASA Ames)

- Highly autonomous systems typically perform numerous *concurrent* activities, e.g., science observations, instrument calibration, fault monitoring & diagnosis, activity planning, etc.
- Mission systems built upon MDS will employ multi-threaded execution, with an *enormous* space of possible states and paths through those states.
- Concurrent interacting programs are particularly vulnerable to *synchronization bugs* such as race conditions and deadlocks.
- ARC is applying and developing analytic verification technology (a.k.a. "model checking") to mathematically analyze specifications, code, and models for consistency with requirements and designs:
  - At JPL, for Mission Data System (D. Dvorak)
  - At GSFC, for the Advanced Architectures & Agents Group (J. Breed)

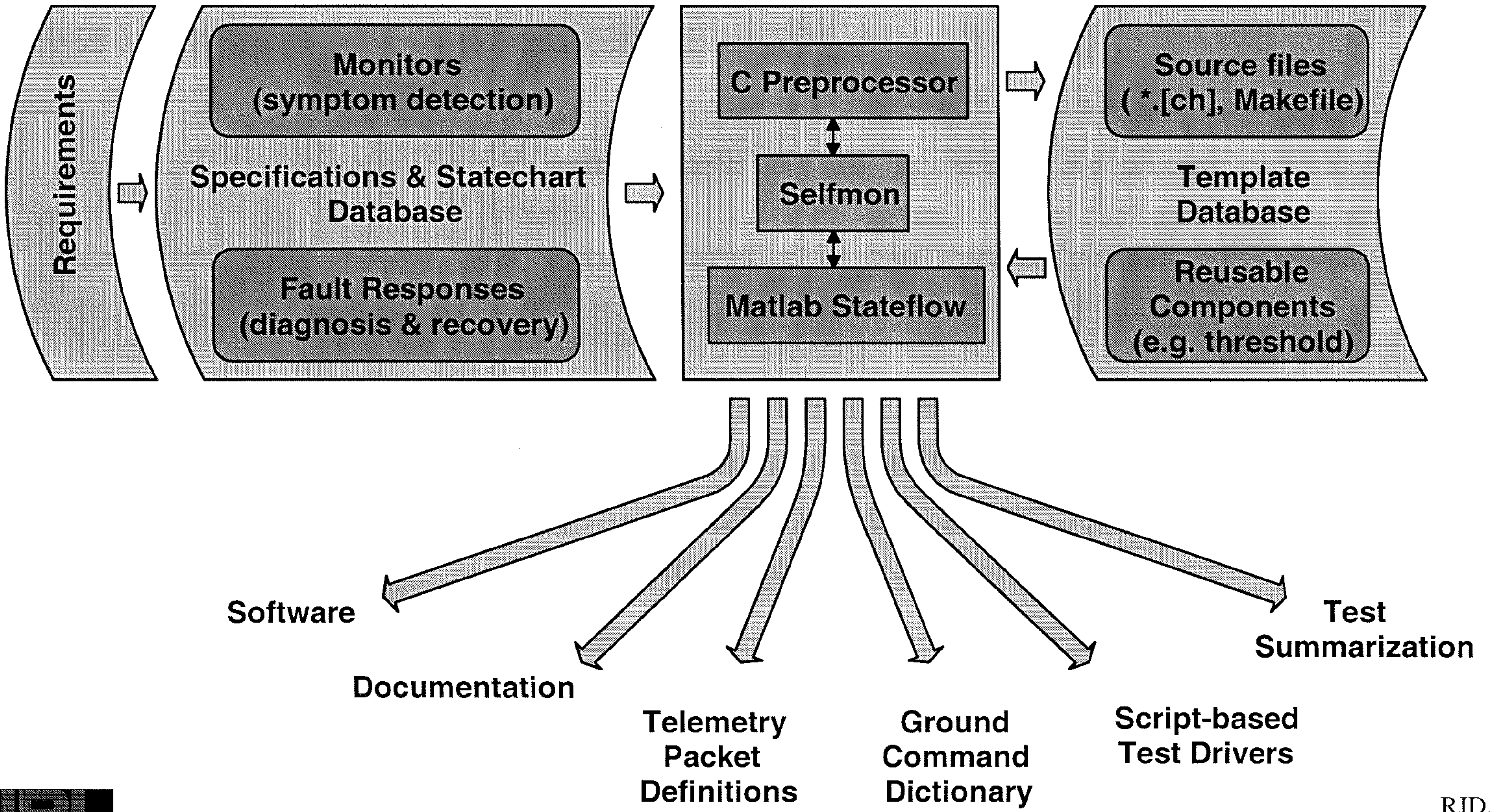
Software Process Technology

Products

A software methodology and process tools for automating software engineering processes including: code generation, documentation, command/telemetry interface, code reuse, and testing via requirement & behavior reconstruction

Benefits

Drastic reduction in software "coding" effort. Simplified interface among organizations such as fault protection users (system engineering, I&T, ground ops), designers, and software engineers


Requirements
Monitors (symptom detection)
Specifications & Statechart Database
Fault Responses (diagnosis & recovery)
C Preprocessor
Selfmon
Matlab Stateflow
Source files (*.[ch], Makefile)
Template Database
Reusable Components (e.g. threshold)
Software
Documentation
Telemetry Packet Definitions
Ground Command Dictionary
Script-based Test Drivers
Test Summarization

Autonomy and Software Engineering

- New critical-path challenges for software engineering are entailed by the autonomy capabilities required for many future NASA missions:
  - Verification and validation
  - Reliability
  - Flight / ground architectures
  - Technologies such as auto-code generation
- The specific drivers emerging from the autonomy area are part of a general pattern of increased importance of software engineering to achieve quality mission software